What's Next for the UML?

With better XML support on the way and open-source modeling tools entering the picture, the relatively quiet Unified Modeling Language world is experiencing major upheaval

By Roger Smith

At a book signing in the mid-1980s, not long after I got out of school with my newly minted computer science degree, I ran into E.F. Codd, the IBM researcher who originated the relational database model. As you might expect, Codd was none too enthusiastic about the term and practice of "object-oriented" development, just then being popularized by methodologists such as Grady Booch, who went on to create (with Jim Rumbaugh and Ivar Jacobsen) the Unified Modeling Language (UML). "You might as well describe birds as 'air-oriented' creatures," Codd complained.

But in the past decade and a half, object technology has been in the ascendancy. Database administrators, who ruled the mainframe and client/server development roost, have been forced to increasingly share power in IT shops as developers have scrambled to adopt object-oriented (OO) languages such as C++ and Java. For the past few years, object and data professionals have brokered an uneasy truce, especially with respect to technology such as Enterprise JavaBeans (EJBs) that are written in Java but typically stored in relational databases such as Oracle or IBM's DB2, which use non-object technology.

To overcome the well-known impedance mismatch problem that exists in relational DBMSs, where an application programmer is forced to work in a language (such as Java or C++) that has a syntax, semantics, and type system different from the data manipulation language (that is, SQL) of the DBMS, many developers have resorted to using some sort of object-relational mapping. Author and consultant Scott Ambler has done the majority of the heavy lifting in this area, with his advocacy of a vendor-neutral UML Persistence Model profile defined as part of the upcoming UML 2.0 standard. (For background, see his white paper "Mapping Objects to Relational Databases: What You Need to Know and Why," posted at IBM DeveloperWorks, or Software Development magazine's "Thinking Objectively" columns.

The EJB UML mapping

A profile is defined as a specialized use of UML within the UML specification, and some of the more interesting work currently going on inside the Java Community Process (the Sun-led participatory process that develops and revises Java technology specifications) revolves around the creation of an EJB UML-mapping profile. The JSR-000026 UML profile for EJB defines a set of extensions to UML that can be used to model software implemented with Enterprise JavaBeans in UML. These extensions will let Java IDE, app server and other enterprise tool vendors provide EJB modeling capabilities using UML within their tools, as well as forward and reverse engineering between UML models and EJB implementations. The specification defines an XML DTD

for a file placed within the EJB-JAR that identifies a UML model stored in that EJB-JAR and its relationship to other EJBs in the same EJB-JAR. This will allow enterprise tool and framework vendors to use Java's automation and reflection APIs to access UML models stored in EJB-JARs.

What is especially compelling about this is that it gives EJB components the capability of self-describing their contents and capabilities, using either use case or other UML diagrams. The proposed profile will also support Extensible Meta-Data Interchange (XMI), the widely used meta-data representation format based on XML.

Tool Support

With Windows UML modeling tool prices running upwards of $3,000 per user (with an additional $1,000 to $2,000 surcharge on the UNIX platform), it's arguably true that the high cost of quality modeling tools has kept the majority of developers from adopting object-oriented analysis and design techniques—which should already have been state of the practice five or six years ago, if not more. But driven by open-source modeling tool efforts such as Thorn and ArgoUML, this state of affairs is about to change. Thorn is a UML modeling tool written in Java that allows you to use XML to save the models you create. The purpose of the Thorn modeling tool is to help develop and manage increasingly sophisticated open source development efforts.

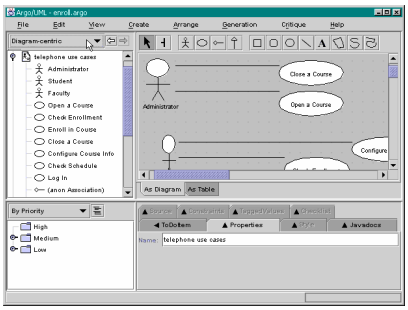

Figure 1 | Use-case diagram built as part of the open-source Enroll.argo project tutorial.

ArgoUML is a modular and extensible open-source Java/UML project from Tigris.org, a mid-sized open-source community that focuses on building better tools for collaborative software development. Based on the UML 1.3 specification and licensed similar to the Apache web server, ArgoUML provides comprehensive support for XMI, the XML Model Interchange format, and OCL (the Object Constraint Language). Developed by Jos Warmer as a language for business modeling within IBM, OCL is a subset of UML that allows software developers to create highly specific sets of rules that govern aspects of an individual object. Since many software projects these days require unique and complex rules written specifically for business models, the ability to write constraints over object models using OCL is particularly useful.

Several UML tool vendors are now climbing on the collaboration tool bandwagon, including market leader Rational, with its Rational Suite set of software development lifecycle tools that integrates the Rational Rose platform with its ClearCase change management solution and a raft of other requirements (RequisitePro), testing (QualityArchitect, Purify, Quantify and TestManager), and verification (PureCoverage) tools. Advanced Software's GDPro product, which was recently acquired by Embarcadero Technologies (who make the ER/Studio data-modeling tool), is banking on collaborating with Java IDE users, including the 1 million-plus people who have downloaded Sun's open-source Forte for Java tool.

Demonstrating an upcoming renamed GDPro release called Describe that can be accessed from a tab on the Forte IDE menu bar, Greg Schottland (former CEO and founder of Advanced and current General Manager of Embarcadero's Application Development Tools division) boasted that "this is the end of reverse engineering and

round-tripping as we know it ... Now a UML diagram will be just another view into a Java project. When you change the source, it changes the diagram, and vice versa."

Cittera from newcomer CanyonBlue, Inc. is a pure-play UML collaboration tool that allows developers in various locations to work on modeling projects in real time, using an Internet-based virtual whiteboard environment. (To get more background on these and other UML tools, check the Objects by Design site, which focuses on recent developments in object-oriented design and programming.)

What's Next for the UML?

Because it offers standardization for notation, modeling artifacts, and semantics, the UML has been a wakeup call for developers eager to work at a higher level of abstraction. In addition to EJB component-based and object-oriented software projects, UML is also able to earn its keep in non-OO software development, like procedural or relational database applications. Despite the best efforts of methodologists such as Ambler and vendors like Rational Software in promoting a standardized way to map objects to relational databases, it seems likely that many or most of these efforts are doomed to failure.

Even though database systems have been around for many years, industry analysts say that 80 percent of the electronic data in companies is unstructured (that is, it resides outside of databases, typically in text files). With the explosion of the World Wide Web and the appearance of content-based search engines (such as Google, AltaVista and Yahoo), it is becoming increasing clear that we can access data successfully even in the absence of a database.

Because much Web data is fragmented, it doesn't make a whole lot of sense to store this data in a bunch of RDBMS tables and then access that data with relational connectors. And with the XML revolution organizing content in a more structured, more semantically meaningful form than HTML, it is increasing likely that developers will want to take a more responsibility-driven or object-oriented (as opposed to data-centric) approach to their applications.

Because they're no longer willing to put all their eggs in one vulnerable RDBMS basket, it's also likely that developers will increasingly turn to UML to raise the level of abstraction of their new Web-based systems, especially if those systems are built of societies of collaborating objects. Although the relational model was a powerful and useful concept in its time, the immediate future will all be about objects and Internet-based UML applications and tools.

Roger Smith *is former technical editor of* Software Development *magazine. He has been a software developer for more than 15 years.*